**在项目中学到的知识点总结**

# [vue $refs的基本用法](http://www.cnblogs.com/xueweijie/p/6907676.html)

<div id="app">

<input type="text" ref="input1"/>

<button @click="add">添加</button>

</div>

[复制代码](javascript:void(0);)

<script>

new Vue({

el: "#app",

methods:{

add:function(){

this.$refs.input1.value ="22"; //this.$refs.input1 减少获取dom节点的消耗

}

}

})

</script>

[复制代码](javascript:void(0);)

一般来讲，获取DOM元素，需document.querySelector（".input1"）获取这个dom节点，然后在获取input1的值。

但是用ref绑定之后，我们就不需要在获取dom节点了，直接在上面的input上绑定input1，然后$refs里面调用就行。

然后在javascript里面这样调用：this.$refs.input1  这样就可以减少获取dom节点的消耗了

# Object.assign的使用

Object.assign方法用于将多个对象的属性复制到一个对象中.

## 语法

Object.assign(target, ...sources)

### 参数

#### target

目标对象, sources和target对象属性进行合并, 复制结束之后, 将结果放到target对象中

#### sources

需要复制属性的对象, 可以是多个

#### Return value

复制返回结果, 和target值一致

### 举例

var obj = { a: 1 };

var copy = Object.assign({}, obj);

console.log(copy); // { a: 1 }

var o1 = { a: 1 };

var o2 = { b: 2 };

var o3 = { c: 3 };

var obj = Object.assign(o1, o2, o3);

console.log(obj); // { a: 1, b: 2, c: 3 }

console.log(o1); // { a: 1, b: 2, c: 3 }, target object itself is changed.

# :visible.sync

这是新版本vue才有的属性，作用就是控制visible显示隐藏：

:visible.sync="dialogTableVisible"

# 转换成 年-月-日 时间格式

**\*\*\*传给后台的都是传毫秒数**

// new Date()获取当前时间

// 可以通过new Date(放要转换的时间元素).getFullYear()获取年份

// 可以通过new Date(放要转换的时间元素).getMonth()+1获取选择时间的月份

// 可以通过new Date(放要转换的时间元素).getDate()获取选择时间的日

// 可以通过new Date(放要转换的时间元素).getDay()获取选择时间的星期

// 可以通过new Date(放要转换的时间元素).getTime()获取选择时间的毫秒数

# Vue2.0---将页面中表格数据导出excel

项目中将后台返回的数据v-for到表格中，然后需要将这个表格导出为EXCEL

一、需要安装三个依赖：

       npm install -S file-saver xlsx

       npm install -D script-loader

二、

项目中新建一个文件夹：（vendor---名字任取），里面放置两个文件Blob.js和 Export2Excel.js。百度可下载。

三、

在.vue文件中写这两个方法：其中list是表格的内容  
export2Excel() {  
　　require.ensure([], () => {  
　　　　const { export\_json\_to\_excel } = require('../../vendor/Export2Excel');  
　　　　const tHeader = ['序号', 'IMSI', 'MSISDN', '证件号码', '姓名'];  
　　　　const filterVal = ['ID', 'imsi', 'msisdn', 'address', 'name'];  
　　　　const list = this.tableData;  
　　　　const data = this.formatJson(filterVal, list);  
　　　　export\_json\_to\_excel(tHeader, data, '列表excel');  
　　})  
},  
formatJson(filterVal, jsonData) {  
　　return jsonData.map(v => filterVal.map(j => v[j]))  
}

四、

按钮导出调用export2Excel方法

　　注：如果webpack报解析错误：在build----webpack.base.conf.js中resolve的alias加入 'vendor': path.resolve(\_\_dirname, '../src/vendor'),即可解决。alias是配置别名

　　Blob.js和 Export2Excel.js文件：链接: http://pan.baidu.com/s/1nvSdw85 密码: 5qhi

# Elementui中的el-Input的auto-complete属性:

input 的属性autocomplete 默认为on，其含义代表是否让浏览器自动记录之前输入的值。很多时候，需要对客户的资料进行保密，防止浏览器软件或者恶意插件获取到，可以在input中加入autocomplete="off" 来关闭记录，系统需要保密的情况下可以使用此参数。

# Elementui中的表单el-form的验证方法:

1. el-form需要绑定验证规则 : rules=“规则名”
2. el-form需要绑定数据对象：model=”数据对象”
3. el-form-item需要绑定prop属性，指向需要验证的属性 prop=”password” (指向需要验证的密码)
4. 组件实例中的data（）{}函数里需要定义验证方法分别对应需要验证的属性，例如验证密码的方法：

data(){

var validatePass = (rule, value, callback) => {

if (value === '') {

callback(new Error('请输入密码'));

} else {

if (this.ruleForm2.checkPass !== '') {

this.$refs.ruleForm2.validateField('checkPass');

}

callback();

}

};

}

1. 在returns{}里设置数据对象和规则
2. 规则里包括prop指向的属性，其实也是数据对象里需要验证的属性。属性里可以包括多个对象，这个对象是用来设置属性对应的验证方法和触发方法的：

规则名：{

password[

{ validator: validatePass, trigger: 'blur' }

]

}

验证密码不能为空完成！

# Element-ui面包屑导航

# Mint UI \*引用组件的两种方式:

1. 在main.js文件中全局引入，在各个组件模块中用==this.$组件名== 去调用

2. 在main.js文件中全局引入，在各个模块中再单独用import引入，然后在js代码中可以直接用==组件名.==去调用了

## \*mintui的headder组件:

## 子组件必须要用slot属性设置位置，而且该属性只有两个值：left和right。

# \* npm介绍与cnpm介绍:

npm介绍

1. 说明：npm（node package manager）是nodejs的包管理器，用于node插件管理（包括安装、卸载、管理依赖等）
2. 使用npm安装插件：命令提示符执行npm install <name> [-g] [--save-dev]   
   <name>：node插件名称。   
   例：npm install gulp-less --save-dev

-g：全局安装。 将会安装在C:\Users\Administrator\AppData\Roaming\npm并且写入系统环境变量； 非全局安装：将会安装在当前定位目录； 全局安装可以通过命令行在任何地方调用它，本地安装将安装在定位目录的node\_modules文件夹下，通过require()调用；

--save：将保存配置信息至package.json（package.json是nodejs项目配置文件）；

-dev：保存至package.json的devDependencies节点，不指定-dev将保存至dependencies节点；

为什么要保存至package.json？因为node插件包相对来说非常庞大，所以不加入版本管理，将配置信息写入package.json并将其加入版本管理，其他开发者对应下载即可（命令提示符执行npm install，则会根据package.json下载所有需要的包）。

3.使用npm卸载插件：npm uninstall <name> [-g] [--save-dev]   
PS：不要直接删除本地插件包   
4.使用npm更新插件：npm update <name> [-g] [--save-dev]   
5.更新全部插件：npm update [--save-dev]   
6.查看npm帮助：npm help   
7.查看当前目录已安装插件：npm list

PS：npm安装插件过程：从[http://registry.npmjs.org](http://registry.npmjs.org/" \t "_blank)下载对应的插件包（该网站服务器位于国外，所以经常下载缓慢或出现异常），解决办法往下看↓↓↓↓↓↓。

选装cnpm

1. 说明：因为npm安装插件是从国外服务器下载，受网络影响大，可能出现异常，如果npm的服务器在中国就好了，所以我们乐于分享的淘宝团队干了这事。来自官网：“这是一个完整 npmjs.org 镜像，你可以用此代替官方版本(只读)，同步频率目前为 10分钟 一次以保证尽量与官方服务同步。”
2. 官方网址：[http://npm.taobao.org](http://npm.taobao.org/" \t "_blank)
3. 安装：命令提示符执行npm install cnpm -g --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/)
4. 注意：安装完后最好查看其版本号cnpm -v或关闭命令提示符重新打开，安装完直接使用有可能会出现错误

注：cnpm跟npm用法完全一致，只是在执行命令时将npm改为cnpm。

# 移动端自适应布局----rem字体

一.px,em,rem之间联系与区别

px：像素是相对于显示器屏幕分辨率而言的相对长度单位，不会受屏幕宽度改变

em：继承父级的font-size，与父级的字体大小有关

rem：是em的升级版，rem只会相对html的值，不会受到父级的影响。

二.移动端怎么用rem

假设设计师的视觉稿是按照iPhone6的宽度来设计的，即375px   
那么，我们可以完全按照视觉稿上的尺寸来赋值给元素的样式，比如视觉稿上的尺寸是80px，那么在css中就可以直接定义width:80px; 页面中所有的尺寸都这样来设置。  
  
当所有的网站所有的页面样式都设置好之后。  
  
1. 设置页面的rem大小  
```css  
  
html {  
font-size: calc(100vw/3.75);  
}

或者（ipone6举例）

html{

font-size:26.67vw;

}

```  
100vw是设备的宽度，除以3.75可以让1rem的大小在iPhone6下等于100px  
  
2. 替换页面中的单位，把所有的px单位替换成rem，除以100，比如前面的80px，就是0.8rem  
这样在iPhone6下，所有元素的尺寸还是和视觉稿的尺寸一样，而iphone5中，因为设备的宽度变小了，100vw/3.75得到的值，会相应的变小，即rem的单位值会变小，页面中所有的尺寸会等比例缩放。  
  
这样就可以做到针对任何分辨率的设备保持视觉一致了。  
最后，前面用到vw单位，但是低版本的设备可能不支持，那么就需要js来处理一下：  
```javascript  
  
document.documentElement.style.fontSize = window.innerWidth/3.75 + 'px'

```

当然，rem也可以用在定宽高等